**Test Scenarios**

TC\_001 - Verify that the logo is present on the page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the home page.

2. Check for the presence of the logo.

Expected Result: The logo should be visible on the page.

TC\_002 - Verify that the logo is clickable and redirects correctly

Pre-condition: The logo is present on the home page.

Steps:

1. Navigate to the home page.

2. Click on the logo.

3. Observe the page redirection.

Expected Result: The logo should be clickable and redirect to the correct page (usually the home page).

TC\_003 - Verify the behaviour when the logo is not clickable

Pre-condition: The logo is present on the home page but is not supposed to be clickable.

Steps:

1. Navigate to the home page.

2. Attempt to click on the logo.

Expected Result: The logo should not be clickable, and there should be no redirection or clickable feedback.

TC\_004 - Verify the page without a logo

Pre-condition: The website is accessible, and the logo is removed or hidden.

Steps:

1. Navigate to the home page.

2. Check for the absence of the logo.

Expected Result: The logo should not be visible on the page, and the page layout should be checked for any issues due to the absence of the logo.

TC\_005 - Verify that the user name is present on the page

Pre-condition: The user is logged in and the website is accessible.

Steps:

1. Navigate to the home page.

2. Check for the presence of the user name in the header.

Expected Result: The user name should be visible on the page.

TC\_006 - Verify that the drop-down menu is available when the user name is clicked

Pre-condition: The user is logged in and the user name is present on the home page.

Steps:

1. Navigate to the home page.

2. Click on the user name in the header.

3. Observe the appearance of the drop-down menu.

Expected Result: The drop-down menu should appear when the user name is clicked.

TC\_007 - Verify that the drop-down menu contains ""View Profile,"" ""Change Password,"" and ""Logout"" options

Pre-condition: The drop-down menu is available when the user name is clicked.

Steps:

1. Navigate to the home page.

2. Click on the user name in the header.

3. Check the options in the drop-down menu.

Expected Result: The drop-down menu should contain ""View Profile,"" ""Change Password,"" and ""Logout"" options.

TC\_008 - Verify the behavior when the drop-down menu does not contain all the expected options

Pre-condition: The drop-down menu is available when the user name is clicked.

Steps:

1. Navigate to the home page.

2. Click on the user name in the header.

3. Check the options in the drop-down menu.

Expected Result: If any expected options (""View Profile,"" ""Change Password,"" or ""Logout"") are missing, an error should be logged or an appropriate message should be displayed.

TC\_009 - Verify the behavior when the drop-down menu does not appear on clicking the user name

Pre-condition: The user name is present on the home page.

Steps:

1. Navigate to the home page.

2. Click on the user name in the header.

Expected Result: If the drop-down menu does not appear, an error should be logged or an appropriate message should be displayed.

TC\_010 - Verify the behavior when the user name is not present on the page

Pre-condition: The user is logged in and the website is accessible.

Steps:

1. Navigate to the home page.

2. Check for the presence of the user name in the header.

Expected Result: If the user name is not present, an error should be logged or an appropriate message should be displayed.

TC\_011 - Verify that the collapse sidebar menu button is present

Pre-condition: The website is accessible.

Steps:

1. Navigate to the home page.

2. Check for the presence of the collapse sidebar menu button.

Expected Result: The collapse sidebar menu button should be visible on the page.

TC\_012 - Verify that clicking the collapse sidebar menu button opens the sidebar

Pre-condition: The collapse sidebar menu button is present on the home page.

Steps:

1. Navigate to the home page.

2. Click on the collapse sidebar menu button.

3. Observe the sidebar opening.

Expected Result: The sidebar should open when the collapse sidebar menu button is clicked.

TC\_013 - Verify the behavior when the collapse sidebar menu button is missing

Pre-condition: The website is accessible.

Steps:

1. Navigate to the home page.

2. Check for the presence of the collapse sidebar menu button.

Expected Result: If the collapse sidebar menu button is missing, an error should be logged or an appropriate message should be displayed.

TC\_014 - Verify the sidebar remains functional after collapsing and reopening

Pre-condition: The collapse sidebar menu button is present on the home page.

Steps:

1. Navigate to the home page.

2. Click on the collapse sidebar menu button to collapse the sidebar.

3. Click on the button again to reopen the sidebar.

4. Interact with the sidebar menu items.

Expected Result: The sidebar should remain functional after collapsing and reopening.

TC\_015 - Verify the behavior when the collapse sidebar menu button is clicked multiple times quickly

Pre-condition: The collapse sidebar menu button is present on the home page.

Steps:

1. Navigate to the home page.

2. Click on the collapse sidebar menu button multiple times quickly.

Expected Result: The sidebar should handle multiple quick clicks gracefully, without any UI glitches or performance issues.

TC\_016 - Verify that the title is present on the page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the home page.

2. Check for the presence of the title.

Expected Result: The title should be visible on the page.

TC\_017 - Verify that the title is correctly styled

Pre-condition: The title is present on the home page.

Steps:

1. Navigate to the home page.

2. Check the styling of the title (font size, color, alignment, etc.).

Expected Result: The title should have the correct styling as per the design specifications.

TC\_018 - Verify the behavior when the title is missing

Pre-condition: The website is accessible, and the title is removed or hidden.

Steps:

1. Navigate to the home page.

2. Check for the presence of the title.

Expected Result: If the title is missing, an error should be logged or an appropriate message should be displayed, and the page layout should be checked for any issues due to the absence of the title.

TC\_019 - Verify the presence of the ""Add"" button in page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the main page.

2. Check for the presence of the ""Add"" button.

Expected Result: The ""Add"" button should be visible on the page.

TC\_020 - Verify that clicking the add button opens the add pop-up correctly

Pre-condition: The ""Add"" button is present on the main page.

Steps:

1. Navigate to the main page.

2. Click on the ""Add"" button.

3. Observe the appearance of the add pop-up.

Expected Result: The add pop-up should open correctly.

TC\_021 - Verify that the add pop-up closes and returns to the main page upon cancellation

Pre-condition: The add pop-up is open.

Steps:

1. Click on the cancel button or close icon on the add pop-up.

2. Observe the behavior.

Expected Result: The add pop-up should close, and the user should return to the main page.

TC\_022 - Verify that the add pop-up opens with correct fields

Pre-condition: The ""Add"" button is present on the main page.

Steps:

1. Navigate to the main page.

2. Click on the ""Add"" button.

3. Observe the fields in the add pop-up.

Expected Result: The add pop-up should open with the correct fields as per the design specifications.

TC\_023 - Ensure that submitting the form successfully adds a new entry with a unique ID

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Fill in the form with valid data.

2. Click on the save button.

3. Observe the new entry in the main page list.

Expected Result: A new entry with a unique ID should be added to the main page list.

TC\_024 - Ensure that submitting the form without unique data does not add a new entry

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Fill in the form with non-unique data.

2. Click on the save button.

3. Observe the behavior.

Expected Result: The form submission should not add a new entry, and an appropriate error message should be displayed.

TC\_025 - Ensure that submitting the form with empty fields does not add a new entry

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Leave some or all fields empty.

2. Click on the save button.

3. Observe the behavior.

Expected Result: The form submission should not add a new entry, and an appropriate error message should be displayed.

TC\_026 - Ensure that submitting the form with invalid data does not add a new entry

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Fill in the form with invalid data.

2. Click on the save button.

3. Observe the behavior.

Expected Result: The form submission should not add a new entry, and an appropriate error message should be displayed.

TC\_027 - Verify that clicking the save button with valid data displays a success message

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Fill in the form with valid data.

2. Click on the save button.

3. Observe the success message.

Expected Result: A success message should be displayed upon saving with valid data.

TC\_028 - Verify that clicking the save button with invalid data does not display a success message

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Fill in the form with invalid data.

2. Click on the save button.

3. Observe the behavior.

Expected Result: No success message should be displayed upon saving with invalid data, and an appropriate error message should be shown.

TC\_029 - Verify that clicking the reset button clears the data entered in all fields

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Fill in the form with any data.

2. Click on the reset button.

3. Observe the fields in the add pop-up.

Expected Result: All fields in the add pop-up should be cleared.

TC\_030 - Verify the behavior when clicking the reset button with no data entered

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Ensure no data is entered in the form fields.

2. Click on the reset button.

3. Observe the behavior.

Expected Result: No change should occur, as there was no data to reset.

TC\_031 - Verify that clicking the reset button does not clear any data

Pre-condition: The add pop-up is open with correct fields.

Steps:

1. Fill in the form with any data.

2. Click on the reset button.

3. Observe the behavior.

Expected Result: All fields in the add pop-up should be cleared.

TC\_032 - Verify that clicking the back button redirects to the main page successfully

Pre-condition: The add pop-up is open.

Steps:

1. Click on the back button.

2. Observe the behavior.

Expected Result: The user should be redirected to the main page.

TC\_033 - Verify that ID is present in the page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the main page.

2. Check for the presence of the ID column.

Expected Result: The ID column should be visible on the page.

TC\_034 - Verify that ID contains sort functionality

Pre-condition: The ID column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the sort option in the ID column.

Expected Result: The ID column should contain a sort option.

TC\_035 - Verify that ID contains filter functionality

Pre-condition: The ID column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the filter option in the ID column.

Expected Result: The ID column should contain a filter option.

TC\_036 - Verify the sort functionality in ID with valid data

Pre-condition: The ID column contains sort functionality.

Steps:

1. Sort the ID column with valid data.

2. Observe the sorting behavior.

Expected Result: The ID column should be sorted correctly with valid data.

TC\_037 - Verify the filter functionality in ID with valid data

Pre-condition: The ID column contains filter functionality.

Steps:

1. Filter the ID column with valid data.

2. Observe the filtering behavior.

Expected Result: The ID column should be filtered correctly with valid data.

TC\_038 - Verify the sort functionality in ID with invalid data

Pre-condition: The ID column contains sort functionality.

Steps:

1. Sort the ID column with invalid data.

2. Observe the sorting behavior.

Expected Result: The ID column should handle invalid data gracefully and provide an appropriate message.

TC\_039 - Verify the filter functionality in ID with invalid data

Pre-condition: The ID column contains filter functionality.

Steps:

1. Filter the ID column with invalid data.

2. Observe the filtering behavior.

Expected Result: The ID column should handle invalid data gracefully and provide an appropriate message.

TC\_040 - Verify the behavior when ID sort functionality is not working

Pre-condition: The ID column is present on the main page.

Steps:

1. Attempt to sort the ID column.

2. Observe the behavior.

Expected Result: If the sort functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_041 - Verify the behavior when ID filter functionality is not working

Pre-condition: The ID column is present on the main page.

Steps:

1. Attempt to filter the ID column.

2. Observe the behavior.

Expected Result: If the filter functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_042 - Verify that Target is present in the page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the main page.

2. Check for the presence of the Target column.

Expected Result: The Target column should be visible on the page.

TC\_043 - Verify that Target contains sort functionality

Pre-condition: The Target column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the sort option in the Target column.

Expected Result: The Target column should contain a sort option.

TC\_044 - Verify that Target contains filter functionality

Pre-condition: The Target column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the filter option in the Target column.

Expected Result: The Target column should contain a filter option.

TC\_045 - Verify the sort functionality in Target with valid data

Pre-condition: The Target column contains sort functionality.

Steps:

1. Sort the Target column with valid data.

2. Observe the sorting behavior.

Expected Result: The Target column should be sorted correctly with valid data.

TC\_046 - Verify the filter functionality in Target with valid data

Pre-condition: The Target column contains filter functionality.

Steps:

1. Filter the Target column with valid data.

2. Observe the filtering behavior.

Expected Result: The Target column should be filtered correctly with valid data.

TC\_047 - Verify the sort functionality in Target with invalid data

Pre-condition: The Target column contains sort functionality.

Steps:

1. Sort the Target column with invalid data.

2. Observe the sorting behavior.

Expected Result: The Target column should handle invalid data gracefully and provide an appropriate message.

TC\_048 - Verify the filter functionality in Target with invalid data

Pre-condition: The Target column contains filter functionality.

Steps:

1. Filter the Target column with invalid data.

2. Observe the filtering behavior.

Expected Result: The Target column should handle invalid data gracefully and provide an appropriate message.

TC\_049 - Verify the behavior when Target sort functionality is not working

Pre-condition: The Target column is present on the main page.

Steps:

1. Attempt to sort the Target column.

2. Observe the behavior.

Expected Result: If the sort functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_050 - Verify the behavior when Target filter functionality is not working

Pre-condition: The Target column is present on the main page.

Steps:

1. Attempt to filter the Target column.

2. Observe the behavior.

Expected Result: If the filter functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_051 - Verify that Subtarget is present in the page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the main page.

2. Check for the presence of the Subtarget column.

Expected Result: The Subtarget column should be visible on the page.

TC\_052 - Verify that Subtarget contains sort functionality

Pre-condition: The Subtarget column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the sort option in the Subtarget column.

Expected Result: The Subtarget column should contain a sort option.

TC\_053 - Verify that Subtarget contains filter functionality

Pre-condition: The Subtarget column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the filter option in the Subtarget column.

Expected Result: The Subtarget column should contain a filter option.

TC\_054 - Verify the sort functionality in Subtarget with valid data

Pre-condition: The Subtarget column contains sort functionality.

Steps:

1. Sort the Subtarget column with valid data.

2. Observe the sorting behavior.

Expected Result: The Subtarget column should be sorted correctly with valid data.

TC\_055 - Verify the filter functionality in Subtarget with valid data

Pre-condition: The Subtarget column contains filter functionality.

Steps:

1. Filter the Subtarget column with valid data.

2. Observe the filtering behavior.

Expected Result: The Subtarget column should be filtered correctly with valid data.

TC\_056 - Verify the sort functionality in Subtarget with invalid data

Pre-condition: The Subtarget column contains sort functionality.

Steps:

1. Sort the Subtarget column with invalid data.

2. Observe the sorting behavior.

Expected Result: The Subtarget column should handle invalid data gracefully and provide an appropriate message.

TC\_057 - Verify the filter functionality in Subtarget with invalid data

Pre-condition: The Subtarget column contains filter functionality.

Steps:

1. Filter the Subtarget column with invalid data.

2. Observe the filtering behavior.

Expected Result: The Subtarget column should handle invalid data gracefully and provide an appropriate message.

TC\_058 - Verify the behavior when Subtarget sort functionality is not working

Pre-condition: The Subtarget column is present on the main page.

Steps:

1. Attempt to sort the Subtarget column.

2. Observe the behavior.

Expected Result: If the sort functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_059 - Verify the behavior when Subtarget filter functionality is not working

Pre-condition: The Subtarget column is present on the main page.

Steps:

1. Attempt to filter the Subtarget column.

2. Observe the behavior.

Expected Result: If the filter functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_060 - Verify that Incorporation City is present in the page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the main page.

2. Check for the presence of the Incorporation City column.

Expected Result: The Incorporation City column should be visible on the page.

TC\_061 - Verify that Incorporation City contains sort functionality

Pre-condition: The Incorporation City column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the sort option in the Incorporation City column.

Expected Result: The Incorporation City column should contain a sort option.

TC\_062 - Verify that Incorporation City contains filter functionality

Pre-condition: The Incorporation City column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the filter option in the Incorporation City column.

Expected Result: The Incorporation City column should contain a filter option.

TC\_063 - Verify the sort functionality in Incorporation City with valid data

Pre-condition: The Incorporation City column contains sort functionality.

Steps:

1. Sort the Incorporation City column with valid data.

2. Observe the sorting behavior.

Expected Result: The Incorporation City column should be sorted correctly with valid data.

TC\_064 - Verify the filter functionality in Incorporation City with valid data

Pre-condition: The Incorporation City column contains filter functionality.

Steps:

1. Filter the Incorporation City column with valid data.

2. Observe the filtering behavior.

Expected Result: The Incorporation City column should be filtered correctly with valid data.

TC\_065 - Verify the sort functionality in Incorporation City with invalid data

Pre-condition: The Incorporation City column contains sort functionality.

Steps:

1. Sort the Incorporation City column with invalid data.

2. Observe the sorting behavior.

Expected Result: The Incorporation City column should handle invalid data gracefully and provide an appropriate message.

TC\_066 - Verify the filter functionality in Incorporation City with invalid data

Pre-condition: The Incorporation City column contains filter functionality.

Steps:

1. Filter the Incorporation City column with invalid data.

2. Observe the filtering behavior.

Expected Result: The Incorporation City column should handle invalid data gracefully and provide an appropriate message.

TC\_067 - Verify the behavior when Incorporation City sort functionality is not working

Pre-condition: The Incorporation City column is present on the main page.

Steps:

1. Attempt to sort the Incorporation City column.

2. Observe the behavior.

Expected Result: If the sort functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_068 - Verify the behavior when Incorporation City filter functionality is not working

Pre-condition: The Incorporation City column is present on the main page.

Steps:

1. Attempt to filter the Incorporation City column.

2. Observe the behavior.

Expected Result: If the filter functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_069 - Verify that Sector Classification is present in the page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the main page.

2. Check for the presence of the Sector Classification column.

Expected Result: The Sector Classification column should be visible on the page.

TC\_070 - Verify that Sector Classification contains sort functionality

Pre-condition: The Sector Classification column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the sort option in the Sector Classification column.

Expected Result: The Sector Classification column should contain a sort option.

TC\_071 - Verify that Sector Classification contains filter functionality

Pre-condition: The Sector Classification column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the filter option in the Sector Classification column.

Expected Result: The Sector Classification column should contain a filter option.

TC\_072 - Verify the sort functionality in Sector Classification with valid data

Pre-condition: The Sector Classification column contains sort functionality.

Steps:

1. Sort the Sector Classification column with valid data.

2. Observe the sorting behavior.

Expected Result: The Sector Classification column should be sorted correctly with valid data.

TC\_073 - Verify the filter functionality in Sector Classification with valid data

Pre-condition: The Sector Classification column contains filter functionality.

Steps:

1. Filter the Sector Classification column with valid data.

2. Observe the filtering behavior.

Expected Result: The Sector Classification column should be filtered correctly with valid data.

TC\_074 - Verify the sort functionality in Sector Classification with invalid data

Pre-condition: The Sector Classification column contains sort functionality.

Steps:

1. Sort the Sector Classification column with invalid data.

2. Observe the sorting behavior.

Expected Result: The Sector Classification column should handle invalid data gracefully and provide an appropriate message.

TC\_075 - Verify the filter functionality in Sector Classification with invalid data

Pre-condition: The Sector Classification column contains filter functionality.

Steps:

1. Filter the Sector Classification column with invalid data.

2. Observe the filtering behavior.

Expected Result: The Sector Classification column should handle invalid data gracefully and provide an appropriate message.

TC\_076 - Verify the behavior when Sector Classification sort functionality is not working

Pre-condition: The Sector Classification column is present on the main page.

Steps:

1. Attempt to sort the Sector Classification column.

2. Observe the behavior.

Expected Result: If the sort functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_077 - Verify the behavior when Sector Classification filter functionality is not working

Pre-condition: The Sector Classification column is present on the main page.

Steps:

1. Attempt to filter the Sector Classification column.

2. Observe the behavior.

Expected Result: If the filter functionality is not working, an error should be logged or an appropriate message should be displayed.

TC\_078 - Verify that Actions is present in the page

Pre-condition: The website is accessible.

Steps:

1. Navigate to the main page.

2. Check for the presence of the Actions column.

Expected Result: The Actions column should be visible on the page.

TC\_079 - Verify that the action column contains the ""Edit icon"" and ""Delete icon"" options

Pre-condition: The Actions column is present on the main page.

Steps:

1. Navigate to the main page.

2. Check for the presence of the ""Edit icon"" and ""Delete icon"" options in the Actions column.

Expected Result: The Actions column should contain the ""Edit icon"" and ""Delete icon"" options.

TC\_080 - Verify that clicking the ""Edit icon"" opens the edit form

Pre-condition: The website is accessible and the table contains rows with an ""Edit icon"".

Steps:

1. Navigate to the main page.

2. Click on the ""Edit icon"" in a table row.

Expected Result: The edit form should open correctly.

TC\_081 - Verify that clicking the edit icon opens the edit form with correct data pre-filled

Pre-condition: The website is accessible and the table contains rows with an ""Edit icon"".

Steps:

1. Navigate to the main page.

2. Click on the ""Edit icon"" in a table row.

3. Observe the pre-filled data in the edit form.

Expected Result: The edit form should open with the correct data pre-filled.

TC\_082 - Verify that clicking the edit icon opens the edit form with some data fields empty

Pre-condition: The website is accessible and the table contains rows with an ""Edit icon"".

Steps:

1. Navigate to the main page.

2. Click on the ""Edit icon"" in a table row with some empty fields.

3. Observe the data in the edit form.

Expected Result: The edit form should open with the correct data pre-filled, including empty fields as they were.

TC\_083 - Verify that clicking the edit icon opens the edit form and allows data modification

Pre-condition: The website is accessible and the table contains rows with an ""Edit icon"".

Steps:

1. Navigate to the main page.

2. Click on the ""Edit icon"" in a table row.

3. Modify the data in the edit form.

4. Submit the form.

Expected Result: The edit form should allow data modification and submit the changes.

TC\_084 - Modify data in the edit form and verify changes are reflected in the table after submission

Pre-condition: The website is accessible and the table contains rows with an ""Edit icon"".

Steps:

1. Navigate to the main page.

2. Click on the ""Edit icon"" in a table row.

3. Modify the data in the edit form.

4. Submit the form.

5. Observe the changes reflected in the table.

Expected Result: The modified data should be accurately reflected in the table after submission.

TC\_085 - Verify that clicking the delete icon shows a deletion prompt

Pre-condition: The website is accessible and the table contains rows with a ""Delete icon"".

Steps:

1. Navigate to the main page.

2. Click on the ""Delete icon"" in a table row.

Expected Result: A deletion prompt should appear.

TC\_086 - Verify that clicking cancel in the deletion prompt closes the dialog without deleting the row

Pre-condition: The deletion prompt is displayed.

Steps:

1. Click on the ""Cancel"" button in the deletion prompt.

2. Observe the behavior.

Expected Result: The deletion prompt should close, and the row should remain in the table.

TC\_087 - Verify that clicking confirm in the deletion prompt deletes the row and shows a success message

Pre-condition: The deletion prompt is displayed.

Steps:

1. Click on the ""Confirm"" button in the deletion prompt.

2. Observe the behavior.

Expected Result: The row should be deleted from the table, and a success message should be displayed.

TC\_088 - Verify that clicking confirm in the deletion prompt fails to delete the row

Pre-condition: The deletion prompt is displayed.

Steps:

1. Simulate a scenario where the deletion action fails (e.g., server error).

2. Click on the ""Confirm"" button in the deletion prompt.

3. Observe the behavior.

Expected Result: The row should not be deleted from the table, and an appropriate error message should be displayed."

TC\_089 - Navigate to the next page using pagination and confirm entries are displayed correctly

Pre-condition: The website is accessible, and pagination is present.

Steps:

1. Navigate to the main page.

2. Click on the ""Next"" button in the pagination controls.

3. Observe the entries displayed on the next page.

Expected Result: Entries on the next page should be displayed correctly.

TC\_090 - Navigate to the previous page using pagination and confirm entries are displayed correctly

Pre-condition: The website is accessible, and pagination is present.

Steps:

1. Navigate to the main page.

2. Click on the ""Next"" button to navigate to the next page.

3. Click on the ""Previous"" button in the pagination controls.

4. Observe the entries displayed on the previous page.

Expected Result: Entries on the previous page should be displayed correctly.

TC\_091 - Navigate to a specific page using pagination and confirm entries are displayed correctly

Pre-condition: The website is accessible, and pagination is present.

Steps:

1. Navigate to the main page.

2. Click on a specific page number in the pagination controls.

3. Observe the entries displayed on the specified page.

Expected Result: Entries on the specified page should be displayed correctly.

TC\_092 - Navigate using pagination controls and confirm correct number of entries per page

Pre-condition: The website is accessible, and pagination is present.

Steps:

1. Navigate to the main page.

2. Observe the number of entries displayed per page.

3. Navigate through several pages using the pagination controls.

Expected Result: The correct number of entries should be displayed on each page.

TC\_093 - Confirm that the entry count updates dynamically when a filter is applied

Pre-condition: The website is accessible, and filters are available.

Steps:

1. Navigate to the main page.

2. Apply a valid filter.

3. Observe the entry count.

Expected Result: The entry count should update dynamically to reflect the filtered entries.

TC\_094 - Confirm that the entry count updates dynamically when a filter is removed

Pre-condition: The website is accessible, and filters are available.

Steps:

1. Navigate to the main page.

2. Apply a valid filter.

3. Remove the filter.

4. Observe the entry count.

Expected Result: The entry count should update dynamically to reflect the original entries.

TC\_095 - Confirm that the entry count does not update when applying an invalid filter

Pre-condition: The website is accessible, and filters are available.

Steps:

1. Navigate to the main page.

2. Apply an invalid filter.

3. Observe the entry count.

Expected Result: The entry count should not update, and an appropriate message should be displayed.

TC\_096 - Confirm that the entry count updates dynamically when multiple filters are applied

Pre-condition: The website is accessible, and filters are available.

Steps:

1. Navigate to the main page.

2. Apply multiple valid filters.

3. Observe the entry count.

Expected Result: The entry count should update dynamically to reflect the entries matching the multiple filters.

TC\_097 - Ensure that the buttons match the application's style guide

Pre-condition: The website is accessible and the style guide is available.

Steps:

1. Navigate to the main page.

2. Inspect the buttons on the page.

3. Compare button styles (color, size, font, border, etc.) to the style guide.

Expected Result: All buttons should match the application's style guide.

TC\_098 - Ensure that the icons match the application's style guide

Pre-condition: The website is accessible and the style guide is available.

Steps:

1. Navigate to the main page.

2. Inspect the icons on the page.

3. Compare icon styles (size, color, design, placement, etc.) to the style guide.

Expected Result: All icons should match the application's style guide.

TC\_099 - Ensure that the text styles match the application's style guide

Pre-condition: The website is accessible and the style guide is available.

Steps:

1. Navigate to the main page.

2. Inspect the text styles on the page.

3. Compare text styles (font type, size, color, spacing, alignment, etc.) to the style guide.

Expected Result: All text styles should match the application's style guide.

TC\_100 - Verify that the table layout adapts appropriately on a desktop screen size

Pre-condition: The website is accessible on a desktop device.

Steps:

1. Navigate to the main page using a desktop device.

2. Observe the table layout.

Expected Result: The table layout should adapt appropriately to the desktop screen size, maintaining readability and usability.

TC\_101 - Verify that the table layout adapts appropriately on a tablet screen size

Pre-condition: The website is accessible on a tablet device.

Steps:

1. Navigate to the main page using a tablet device.

2. Observe the table layout.

Expected Result: The table layout should adapt appropriately to the tablet screen size, maintaining readability and usability.

TC\_102 - Verify that the table layout adapts appropriately on a mobile screen size

Pre-condition: The website is accessible on a mobile device.

Steps:

1. Navigate to the main page using a mobile device.

2. Observe the table layout.

Expected Result: The table layout should adapt appropriately to the mobile screen size, maintaining readability and usability.

**Test Automation Scripts**

Below are the Selenium test automation scripts for the test cases.

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

import unittest

class TestWebsite(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Chrome() # Make sure you have the appropriate WebDriver installed

def tearDown(self):

self.driver.quit()

def test\_TC\_001\_logo\_present(self):

driver = self.driver

driver.get("http://example.com") # Replace with your website URL

logo = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "logo-selector"))) # Replace 'logo-selector' with the actual CSS selector for the logo

self.assertTrue(logo.is\_displayed(), "Logo should be visible on the page")

def test\_TC\_002\_logo\_clickable(self):

driver = self.driver

driver.get("http://example.com")

logo = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "logo-selector"))) # Replace 'logo-selector' with the actual CSS selector for the logo

logo.click()

self.assertEqual(driver.current\_url, "http://example.com/home", "Logo should redirect to the home page") # Replace with the correct URL

def test\_TC\_003\_logo\_not\_clickable(self):

driver = self.driver

driver.get("http://example.com")

logo = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "logo-selector")))

try:

logo.click()

self.fail("Logo should not be clickable")

except:

pass

def test\_TC\_004\_logo\_absent(self):

driver = self.driver

driver.get("http://example.com")

logos = driver.find\_elements(By.CSS\_SELECTOR, "logo-selector")

self.assertEqual(len(logos), 0, "Logo should not be visible on the page")

def test\_TC\_005\_user\_name\_present(self):

driver = self.driver

driver.get("http://example.com")

user\_name = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "username-selector"))) # Replace 'username-selector' with the actual CSS selector for the user name

self.assertTrue(user\_name.is\_displayed(), "User name should be visible on the page")

def test\_TC\_006\_user\_name\_click\_dropdown(self):

driver = self.driver

driver.get("http://example.com")

user\_name = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "username-selector")))

user\_name.click()

dropdown = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "dropdown-menu-selector"))) # Replace with actual dropdown menu selector

self.assertTrue(dropdown.is\_displayed(), "Dropdown menu should appear")

def test\_TC\_007\_dropdown\_options(self):

driver = self.driver

driver.get("http://example.com")

user\_name = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "username-selector")))

user\_name.click()

dropdown = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "dropdown-menu-selector")))

options = dropdown.find\_elements(By.TAG\_NAME, "li") # Adjust according to your dropdown structure

expected\_options = ["View Profile", "Change Password", "Logout"]

actual\_options = [option.text for option in options]

self.assertEqual(expected\_options, actual\_options, "Dropdown menu should contain the correct options")

def test\_TC\_008\_missing\_dropdown\_options(self):

driver = self.driver

driver.get("http://example.com")

user\_name = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "username-selector")))

user\_name.click()

dropdown = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "dropdown-menu-selector")))

options = dropdown.find\_elements(By.TAG\_NAME, "li")

expected\_options = ["View Profile", "Change Password", "Logout"]

actual\_options = [option.text for option in options]

for option in expected\_options:

self.assertIn(option, actual\_options, f"Option {option} should be present in the dropdown")

def test\_TC\_009\_no\_dropdown\_on\_click(self):

driver = self.driver

driver.get("http://example.com")

user\_name = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "username-selector")))

user\_name.click()

try:

dropdown = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "dropdown-menu-selector")))

self.fail("Dropdown menu should not appear")

except:

pass

def test\_TC\_010\_user\_name\_absent(self):

driver = self.driver

driver.get("http://example.com")

user\_names = driver.find\_elements(By.CSS\_SELECTOR, "username-selector")

self.assertEqual(len(user\_names), 0, "User name should not be visible on the page")

def test\_TC\_011\_collapse\_sidebar\_button\_present(self):

driver = self.driver

driver.get("http://example.com")

button = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "sidebar-button-selector"))) # Replace 'sidebar-button-selector' with the actual CSS selector

self.assertTrue(button.is\_displayed(), "Collapse sidebar menu button should be visible")

def test\_TC\_012\_collapse\_sidebar(self):

driver = self.driver

driver.get("http://example.com")

button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sidebar-button-selector"))) # Replace 'sidebar-button-selector' with the actual CSS selector

button.click()

sidebar = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "sidebar-selector"))) # Replace 'sidebar-selector' with the actual CSS selector

self.assertTrue(sidebar.is\_displayed(), "Sidebar should open when the button is clicked")

def test\_TC\_013\_sidebar\_button\_missing(self):

driver = self.driver

driver.get("http://example.com")

buttons = driver.find\_elements(By.CSS\_SELECTOR, "sidebar-button-selector")

self.assertEqual(len(buttons), 0, "Collapse sidebar menu button should not be visible")

def test\_TC\_014\_sidebar\_functionality(self):

driver = self.driver

driver.get("http://example.com")

button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sidebar-button-selector")))

button.click()

button.click()

sidebar\_items = WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "sidebar-item-selector"))) # Replace 'sidebar-item-selector' with the actual CSS selector

for item in sidebar\_items:

self.assertTrue(item.is\_displayed(), "Sidebar item should be functional")

def test\_TC\_015\_sidebar\_multiple\_clicks(self):

driver = self.driver

driver.get("http://example.com")

button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sidebar-button-selector")))

for \_ in range(5):

button.click()

self.assertTrue(button.is\_displayed(), "Sidebar button should handle multiple clicks without issues")

def test\_TC\_016\_title\_present(self):

driver = self.driver

driver.get("http://example.com")

title = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "title-selector"))) # Replace 'title-selector' with the actual CSS selector for the title

self.assertTrue(title.is\_displayed(), "Title should be visible on the page")

def test\_TC\_017\_title\_styling(self):

driver = self.driver

driver.get("http://example.com")

title = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "title-selector")))

self.assertEqual(title.value\_of\_css\_property('font-size'), 'expected-font-size') # Replace with the expected font size

self.assertEqual(title.value\_of\_css\_property('color'), 'expected-color') # Replace with the expected color

self.assertEqual(title.value\_of\_css\_property('text-align'), 'expected-alignment') # Replace with the expected alignment

def test\_TC\_018\_title\_absent(self):

driver = self.driver

driver.get("http://example.com")

titles = driver.find\_elements(By.CSS\_SELECTOR, "title-selector")

self.assertEqual(len(titles), 0, "Title should not be visible on the page")

def test\_TC\_019\_add\_button\_present(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "add-button-selector"))) # Replace 'add-button-selector' with the actual CSS selector for the add button

self.assertTrue(add\_button.is\_displayed(), "Add button should be visible on the page")

def test\_TC\_020\_add\_button\_functionality(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

popup = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "add-popup-selector"))) # Replace 'add-popup-selector' with the actual CSS selector for the add pop-up

self.assertTrue(popup.is\_displayed(), "Add pop-up should open correctly")

def test\_TC\_021\_add\_popup\_close(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

close\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "close-popup-selector"))) # Replace 'close-popup-selector' with the actual CSS selector for the close button

close\_button.click()

popup = driver.find\_elements(By.CSS\_SELECTOR, "add-popup-selector")

self.assertEqual(len(popup), 0, "Add pop-up should close and return to the main page")

def test\_TC\_022\_add\_popup\_fields(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

fields = WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "add-popup-field-selector"))) # Replace 'add-popup-field-selector' with the actual CSS selector for the fields

self.assertEqual(len(fields), expected\_field\_count) # Replace 'expected\_field\_count' with the number of fields expected

for field in fields:

self.assertTrue(field.is\_displayed(), "Field should be visible in the add pop-up")

def test\_TC\_023\_add\_entry\_unique\_id(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "field-selector"))) # Replace 'field-selector' with the actual CSS selector for the field

field.send\_keys("unique data")

save\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "save-button-selector"))) # Replace 'save-button-selector' with the actual CSS selector for the save button

save\_button.click()

entry = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "entry-selector"))) # Replace 'entry-selector' with the actual CSS selector for the entry

self.assertTrue(entry.is\_displayed(), "New entry with unique ID should be added")

def test\_TC\_024\_add\_entry\_non\_unique(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "field-selector")))

field.send\_keys("non-unique data")

save\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "save-button-selector")))

save\_button.click()

error\_message = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "error-message-selector"))) # Replace 'error-message-selector' with the actual CSS selector for the error message

self.assertTrue(error\_message.is\_displayed(), "Error message should be displayed for non-unique data")

def test\_TC\_025\_add\_entry\_empty\_fields(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

save\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "save-button-selector")))

save\_button.click()

error\_message = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "error-message-selector")))

self.assertTrue(error\_message.is\_displayed(), "Error message should be displayed for empty fields")

def test\_TC\_026\_add\_entry\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "field-selector")))

field.send\_keys("invalid data")

save\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "save-button-selector")))

save\_button.click()

error\_message = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "error-message-selector")))

self.assertTrue(error\_message.is\_displayed(), "Error message should be displayed for invalid data")

def test\_TC\_027\_success\_message\_on\_save(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "field-selector")))

field.send\_keys("valid data")

save\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "save-button-selector")))

save\_button.click()

success\_message = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "success-message-selector"))) # Replace 'success-message-selector' with the actual CSS selector for the success message

self.assertTrue(success\_message.is\_displayed(), "Success message should be displayed upon saving with valid data")

def test\_TC\_028\_no\_success\_message\_on\_invalid\_save(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "field-selector")))

field.send\_keys("invalid data")

save\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "save-button-selector")))

save\_button.click()

error\_message = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "error-message-selector")))

self.assertTrue(error\_message.is\_displayed(), "Error message should be displayed, and no success message should be shown")

def test\_TC\_029\_reset\_button\_clears\_fields(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "field-selector")))

field.send\_keys("any data")

reset\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "reset-button-selector"))) # Replace 'reset-button-selector' with the actual CSS selector for the reset button

reset\_button.click()

self.assertEqual(field.get\_attribute('value'), "", "All fields should be cleared")

def test\_TC\_030\_reset\_button\_no\_change(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

reset\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "reset-button-selector")))

reset\_button.click()

field = driver.find\_element(By.CSS\_SELECTOR, "field-selector")

self.assertEqual(field.get\_attribute('value'), "", "No change should occur as there was no data to reset")

def test\_TC\_031\_reset\_button\_does\_not\_clear(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "field-selector")))

field.send\_keys("any data")

reset\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "reset-button-selector")))

reset\_button.click()

self.assertEqual(field.get\_attribute('value'), "", "All fields should be cleared")

def test\_TC\_032\_back\_button\_redirects(self):

driver = self.driver

driver.get("http://example.com")

add\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "add-button-selector")))

add\_button.click()

back\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "back-button-selector"))) # Replace 'back-button-selector' with the actual CSS selector for the back button

back\_button.click()

self.assertEqual(driver.current\_url, "http://example.com/main", "User should be redirected to the main page") # Replace with the correct URL

def test\_TC\_033\_id\_present(self):

driver = self.driver

driver.get("http://example.com")

id\_column = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "id-column-selector"))) # Replace 'id-column-selector' with the actual CSS selector for the ID column

self.assertTrue(id\_column.is\_displayed(), "ID column should be visible on the page")

def test\_TC\_034\_id\_sort\_functionality(self):

driver = self.driver

driver.get("http://example.com")

id\_sort = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "id-sort-selector"))) # Replace 'id-sort-selector' with the actual CSS selector for the ID sort option

self.assertTrue(id\_sort.is\_displayed(), "ID column should contain a sort option")

def test\_TC\_035\_id\_filter\_functionality(self):

driver = self.driver

driver.get("http://example.com")

id\_filter = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "id-filter-selector"))) # Replace 'id-filter-selector' with the actual CSS selector for the ID filter option

self.assertTrue(id\_filter.is\_displayed(), "ID column should contain a filter option")

def test\_TC\_036\_id\_sort\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

id\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "id-sort-selector")))

id\_sort.click()

sorted\_ids = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "id-column-selector")]

self.assertEqual(sorted\_ids, sorted(sorted\_ids), "ID column should be sorted correctly with valid data")

def test\_TC\_037\_id\_filter\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

id\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "id-filter-selector")))

id\_filter.send\_keys("valid data")

filtered\_ids = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "id-column-selector")]

self.assertTrue(all("valid data" in id for id in filtered\_ids), "ID column should be filtered correctly with valid data")

def test\_TC\_038\_id\_sort\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

id\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "id-sort-selector")))

id\_sort.click()

try:

sorted\_ids = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "id-column-selector")]

self.fail("ID column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_039\_id\_filter\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

id\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "id-filter-selector")))

id\_filter.send\_keys("invalid data")

try:

filtered\_ids = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "id-column-selector")]

self.fail("ID column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_040\_id\_sort\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

id\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "id-sort-selector")))

id\_sort.click()

try:

sorted\_ids = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "id-column-selector")]

self.fail("ID column sort functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_041\_id\_filter\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

id\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "id-filter-selector")))

id\_filter.send\_keys("filter")

try:

filtered\_ids = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "id-column-selector")]

self.fail("ID column filter functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_042\_target\_present(self):

driver = self.driver

driver.get("http://example.com")

target\_column = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "target-column-selector"))) # Replace 'target-column-selector' with the actual CSS selector for the target column

self.assertTrue(target\_column.is\_displayed(), "Target column should be visible on the page")

def test\_TC\_043\_target\_sort\_functionality(self):

driver = self.driver

driver.get("http://example.com")

target\_sort = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "target-sort-selector"))) # Replace 'target-sort-selector' with the actual CSS selector for the target sort option

self.assertTrue(target\_sort.is\_displayed(), "Target column should contain a sort option")

def test\_TC\_044\_target\_filter\_functionality(self):

driver = self.driver

driver.get("http://example.com")

target\_filter = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "target-filter-selector"))) # Replace 'target-filter-selector' with the actual CSS selector for the target filter option

self.assertTrue(target\_filter.is\_displayed(), "Target column should contain a filter option")

def test\_TC\_045\_target\_sort\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

target\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "target-sort-selector")))

target\_sort.click()

sorted\_targets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "target-column-selector")]

self.assertEqual(sorted\_targets, sorted(sorted\_targets), "Target column should be sorted correctly with valid data")

def test\_TC\_046\_target\_filter\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

target\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "target-filter-selector")))

target\_filter.send\_keys("valid data")

filtered\_targets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "target-column-selector")]

self.assertTrue(all("valid data" in target for target in filtered\_targets), "Target column should be filtered correctly with valid data")

def test\_TC\_047\_target\_sort\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

target\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "target-sort-selector")))

target\_sort.click()

try:

sorted\_targets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "target-column-selector")]

self.fail("Target column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_048\_target\_filter\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

target\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "target-filter-selector")))

target\_filter.send\_keys("invalid data")

try:

filtered\_targets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "target-column-selector")]

self.fail("Target column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_049\_target\_sort\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

target\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "target-sort-selector")))

target\_sort.click()

try:

sorted\_targets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "target-column-selector")]

self.fail("Target column sort functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_050\_target\_filter\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

target\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "target-filter-selector")))

target\_filter.send\_keys("filter")

try:

filtered\_targets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "target-column-selector")]

self.fail("Target column filter functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_051\_subtarget\_present(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_column = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "subtarget-column-selector"))) # Replace 'subtarget-column-selector' with the actual CSS selector for the subtarget column

self.assertTrue(subtarget\_column.is\_displayed(), "Subtarget column should be visible on the page")

def test\_TC\_052\_subtarget\_sort\_functionality(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_sort = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "subtarget-sort-selector"))) # Replace 'subtarget-sort-selector' with the actual CSS selector for the subtarget sort option

self.assertTrue(subtarget\_sort.is\_displayed(), "Subtarget column should contain a sort option")

def test\_TC\_053\_subtarget\_filter\_functionality(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_filter = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "subtarget-filter-selector"))) # Replace 'subtarget-filter-selector' with the actual CSS selector for the subtarget filter option

self.assertTrue(subtarget\_filter.is\_displayed(), "Subtarget column should contain a filter option")

def test\_TC\_054\_subtarget\_sort\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "subtarget-sort-selector")))

subtarget\_sort.click()

sorted\_subtargets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "subtarget-column-selector")]

self.assertEqual(sorted\_subtargets, sorted(sorted\_subtargets), "Subtarget column should be sorted correctly with valid data")

def test\_TC\_055\_subtarget\_filter\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "subtarget-filter-selector")))

subtarget\_filter.send\_keys("valid data")

filtered\_subtargets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "subtarget-column-selector")]

self.assertTrue(all("valid data" in subtarget for subtarget in filtered\_subtargets), "Subtarget column should be filtered correctly with valid data")

def test\_TC\_056\_subtarget\_sort\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "subtarget-sort-selector")))

subtarget\_sort.click()

try:

sorted\_subtargets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "subtarget-column-selector")]

self.fail("Subtarget column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_057\_subtarget\_filter\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "subtarget-filter-selector")))

subtarget\_filter.send\_keys("invalid data")

try:

filtered\_subtargets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "subtarget-column-selector")]

self.fail("Subtarget column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_058\_subtarget\_sort\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "subtarget-sort-selector")))

subtarget\_sort.click()

try:

sorted\_subtargets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "subtarget-column-selector")]

self.fail("Subtarget column sort functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_059\_subtarget\_filter\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

subtarget\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "subtarget-filter-selector")))

subtarget\_filter.send\_keys("filter")

try:

filtered\_subtargets = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "subtarget-column-selector")]

self.fail("Subtarget column filter functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_060\_incorporation\_city\_present(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_column = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "incorporation-city-column-selector"))) # Replace 'incorporation-city-column-selector' with the actual CSS selector for the incorporation city column

self.assertTrue(incorporation\_city\_column.is\_displayed(), "Incorporation City column should be visible on the page")

def test\_TC\_061\_incorporation\_city\_sort\_functionality(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_sort = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "incorporation-city-sort-selector"))) # Replace 'incorporation-city-sort-selector' with the actual CSS selector for the incorporation city sort option

self.assertTrue(incorporation\_city\_sort.is\_displayed(), "Incorporation City column should contain a sort option")

def test\_TC\_062\_incorporation\_city\_filter\_functionality(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_filter = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "incorporation-city-filter-selector"))) # Replace 'incorporation-city-filter-selector' with the actual CSS selector for the incorporation city filter option

self.assertTrue(incorporation\_city\_filter.is\_displayed(), "Incorporation City column should contain a filter option")

def test\_TC\_063\_incorporation\_city\_sort\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "incorporation-city-sort-selector")))

incorporation\_city\_sort.click()

sorted\_cities = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "incorporation-city-column-selector")]

self.assertEqual(sorted\_cities, sorted(sorted\_cities), "Incorporation City column should be sorted correctly with valid data")

def test\_TC\_064\_incorporation\_city\_filter\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "incorporation-city-filter-selector")))

incorporation\_city\_filter.send\_keys("valid data")

filtered\_cities = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "incorporation-city-column-selector")]

self.assertTrue(all("valid data" in city for city in filtered\_cities), "Incorporation City column should be filtered correctly with valid data")

def test\_TC\_065\_incorporation\_city\_sort\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "incorporation-city-sort-selector")))

incorporation\_city\_sort.click()

try:

sorted\_cities = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "incorporation-city-column-selector")]

self.fail("Incorporation City column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_066\_incorporation\_city\_filter\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "incorporation-city-filter-selector")))

incorporation\_city\_filter.send\_keys("invalid data")

try:

filtered\_cities = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "incorporation-city-column-selector")]

self.fail("Incorporation City column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_067\_incorporation\_city\_sort\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "incorporation-city-sort-selector")))

incorporation\_city\_sort.click()

try:

sorted\_cities = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "incorporation-city-column-selector")]

self.fail("Incorporation City column sort functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_068\_incorporation\_city\_filter\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

incorporation\_city\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "incorporation-city-filter-selector")))

incorporation\_city\_filter.send\_keys("filter")

try:

filtered\_cities = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "incorporation-city-column-selector")]

self.fail("Incorporation City column filter functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_069\_sector\_classification\_present(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_column = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "sector-classification-column-selector"))) # Replace 'sector-classification-column-selector' with the actual CSS selector for the sector classification column

self.assertTrue(sector\_classification\_column.is\_displayed(), "Sector Classification column should be visible on the page")

def test\_TC\_070\_sector\_classification\_sort\_functionality(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_sort = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "sector-classification-sort-selector"))) # Replace 'sector-classification-sort-selector' with the actual CSS selector for the sector classification sort option

self.assertTrue(sector\_classification\_sort.is\_displayed(), "Sector Classification column should contain a sort option")

def test\_TC\_071\_sector\_classification\_filter\_functionality(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_filter = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "sector-classification-filter-selector"))) # Replace 'sector-classification-filter-selector' with the actual CSS selector for the sector classification filter option

self.assertTrue(sector\_classification\_filter.is\_displayed(), "Sector Classification column should contain a filter option")

def test\_TC\_072\_sector\_classification\_sort\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sector-classification-sort-selector")))

sector\_classification\_sort.click()

sorted\_classifications = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "sector-classification-column-selector")]

self.assertEqual(sorted\_classifications, sorted(sorted\_classifications), "Sector Classification column should be sorted correctly with valid data")

def test\_TC\_073\_sector\_classification\_filter\_valid\_data(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sector-classification-filter-selector")))

sector\_classification\_filter.send\_keys("valid data")

filtered\_classifications = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "sector-classification-column-selector")]

self.assertTrue(all("valid data" in classification for classification in filtered\_classifications), "Sector Classification column should be filtered correctly with valid data")

def test\_TC\_074\_sector\_classification\_sort\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sector-classification-sort-selector")))

sector\_classification\_sort.click()

try:

sorted\_classifications = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "sector-classification-column-selector")]

self.fail("Sector Classification column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_075\_sector\_classification\_filter\_invalid\_data(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sector-classification-filter-selector")))

sector\_classification\_filter.send\_keys("invalid data")

try:

filtered\_classifications = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "sector-classification-column-selector")]

self.fail("Sector Classification column should handle invalid data gracefully and provide an appropriate message")

except:

pass

def test\_TC\_076\_sector\_classification\_sort\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_sort = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sector-classification-sort-selector")))

sector\_classification\_sort.click()

try:

sorted\_classifications = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "sector-classification-column-selector")]

self.fail("Sector Classification column sort functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_077\_sector\_classification\_filter\_not\_working(self):

driver = self.driver

driver.get("http://example.com")

sector\_classification\_filter = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "sector-classification-filter-selector")))

sector\_classification\_filter.send\_keys("filter")

try:

filtered\_classifications = [element.text for element in driver.find\_elements(By.CSS\_SELECTOR, "sector-classification-column-selector")]

self.fail("Sector Classification column filter functionality should log an error or display an appropriate message when not working")

except:

pass

def test\_TC\_078\_actions\_present(self):

driver = self.driver

driver.get("http://example.com")

actions\_column = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "actions-column-selector"))) # Replace 'actions-column-selector' with the actual CSS selector for the actions column

self.assertTrue(actions\_column.is\_displayed(), "Actions column should be visible on the page")

def test\_TC\_079\_actions\_edit\_delete\_present(self):

driver = self.driver

driver.get("http://example.com")

actions\_column = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "actions-column-selector")))

edit\_icon = actions\_column.find\_element(By.CSS\_SELECTOR, "edit-icon-selector") # Replace 'edit-icon-selector' with the actual CSS selector for the edit icon

delete\_icon = actions\_column.find\_element(By.CSS\_SELECTOR, "delete-icon-selector") # Replace 'delete-icon-selector' with the actual CSS selector for the delete icon

self.assertTrue(edit\_icon.is\_displayed(), "Edit icon should be visible in the actions column")

self.assertTrue(delete\_icon.is\_displayed(), "Delete icon should be visible in the actions column")

def test\_TC\_080\_click\_edit\_opens\_form(self):

driver = self.driver

driver.get("http://example.com")

edit\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "edit-icon-selector")))

edit\_icon.click()

edit\_form = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "edit-form-selector"))) # Replace 'edit-form-selector' with the actual CSS selector for the edit form

self.assertTrue(edit\_form.is\_displayed(), "Edit form should open correctly")

def test\_TC\_081\_edit\_form\_correct\_data(self):

driver = self.driver

driver.get("http://example.com")

edit\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "edit-icon-selector")))

edit\_icon.click()

edit\_form = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "edit-form-selector")))

self.assertEqual(edit\_form.find\_element(By.CSS\_SELECTOR, "field-selector").get\_attribute('value'), "expected value") # Replace 'field-selector' and 'expected value' with the actual field selector and expected value

def test\_TC\_082\_edit\_form\_empty\_fields(self):

driver = self.driver

driver.get("http://example.com")

edit\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "edit-icon-selector")))

edit\_icon.click()

edit\_form = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "edit-form-selector")))

empty\_fields = [field for field in edit\_form.find\_elements(By.CSS\_SELECTOR, "field-selector") if field.get\_attribute('value') == ""] # Replace 'field-selector' with the actual field selector

self.assertTrue(len(empty\_fields) > 0, "Edit form should open with some empty fields")

def test\_TC\_083\_edit\_form\_data\_modification(self):

driver = self.driver

driver.get("http://example.com")

edit\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "edit-icon-selector")))

edit\_icon.click()

edit\_form = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "edit-form-selector")))

field = edit\_form.find\_element(By.CSS\_SELECTOR, "field-selector") # Replace 'field-selector' with the actual field selector

field.clear()

field.send\_keys("modified data")

save\_button = edit\_form.find\_element(By.CSS\_SELECTOR, "save-button-selector") # Replace 'save-button-selector' with the actual save button selector

save\_button.click()

updated\_field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "updated-field-selector"))) # Replace 'updated-field-selector' with the actual selector for the updated field

self.assertEqual(updated\_field.text, "modified data", "Edit form should allow data modification and submit the changes")

def test\_TC\_084\_edit\_form\_reflects\_changes(self):

driver = self.driver

driver.get("http://example.com")

edit\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "edit-icon-selector")))

edit\_icon.click()

edit\_form = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "edit-form-selector")))

field = edit\_form.find\_element(By.CSS\_SELECTOR, "field-selector") # Replace 'field-selector' with the actual field selector

field.clear()

field.send\_keys("modified data")

save\_button = edit\_form.find\_element(By.CSS\_SELECTOR, "save-button-selector") # Replace 'save-button-selector' with the actual save button selector

save\_button.click()

updated\_field = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "updated-field-selector"))) # Replace 'updated-field-selector' with the actual selector for the updated field

self.assertEqual(updated\_field.text, "modified data", "Modified data should be accurately reflected in the table after submission")

def test\_TC\_085\_delete\_prompt(self):

driver = self.driver

driver.get("http://example.com")

delete\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "delete-icon-selector")))

delete\_icon.click()

delete\_prompt = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "delete-prompt-selector"))) # Replace 'delete-prompt-selector' with the actual CSS selector for the delete prompt

self.assertTrue(delete\_prompt.is\_displayed(), "A deletion prompt should appear")

def test\_TC\_086\_cancel\_delete\_prompt(self):

driver = self.driver

driver.get("http://example.com")

delete\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "delete-icon-selector")))

delete\_icon.click()

cancel\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "cancel-delete-selector"))) # Replace 'cancel-delete-selector' with the actual CSS selector for the cancel button

cancel\_button.click()

delete\_prompt = driver.find\_elements(By.CSS\_SELECTOR, "delete-prompt-selector")

self.assertEqual(len(delete\_prompt), 0, "Deletion prompt should close, and the row should remain in the table")

def test\_TC\_087\_confirm\_delete(self):

driver = self.driver

driver.get("http://example.com")

delete\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "delete-icon-selector")))

delete\_icon.click()

confirm\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "confirm-delete-selector"))) # Replace 'confirm-delete-selector' with the actual CSS selector for the confirm button

confirm\_button.click()

success\_message = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "success-message-selector"))) # Replace 'success-message-selector' with the actual CSS selector for the success message

self.assertTrue(success\_message.is\_displayed(), "Row should be deleted from the table, and a success message should be displayed")

def test\_TC\_088\_delete\_fail(self):

driver = self.driver

driver.get("http://example.com")

delete\_icon = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "delete-icon-selector")))

delete\_icon.click()

confirm\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "confirm-delete-selector")))

confirm\_button.click()

error\_message = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "error-message-selector"))) # Replace 'error-message-selector' with the actual CSS selector for the error message

self.assertTrue(error\_message.is\_displayed(), "Row should not be deleted from the table, and an appropriate error message should be displayed")

def test\_TC\_089\_pagination\_next\_page(self):

driver = self.driver

driver.get("http://example.com")

next\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "pagination-next-selector"))) # Replace 'pagination-next-selector' with the actual CSS selector for the next button

next\_button.click()

entries = WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector"))) # Replace 'entry-selector' with the actual CSS selector for the entries

self.assertTrue(len(entries) > 0, "Entries on the next page should be displayed correctly")

def test\_TC\_090\_pagination\_previous\_page(self):

driver = self.driver

driver.get("http://example.com")

next\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "pagination-next-selector")))

next\_button.click()

previous\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "pagination-previous-selector"))) # Replace 'pagination-previous-selector' with the actual CSS selector for the previous button

previous\_button.click()

entries = WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector")))

self.assertTrue(len(entries) > 0, "Entries on the previous page should be displayed correctly")

def test\_TC\_091\_pagination\_specific\_page(self):

driver = self.driver

driver.get("http://example.com")

page\_number = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "pagination-page-selector"))) # Replace 'pagination-page-selector' with the actual CSS selector for the specific page number

page\_number.click()

entries = WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector")))

self.assertTrue(len(entries) > 0, "Entries on the specified page should be displayed correctly")

def test\_TC\_092\_pagination\_entry\_count(self):

driver = self.driver

driver.get("http://example.com")

entries = WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector")))

entry\_count\_per\_page = len(entries)

self.assertEqual(entry\_count\_per\_page, expected\_count\_per\_page) # Replace 'expected\_count\_per\_page' with the actual number of entries expected per page

for \_ in range(3):

next\_button = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "pagination-next-selector")))

next\_button.click()

entries = WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector")))

self.assertEqual(len(entries), entry\_count\_per\_page, "The correct number of entries should be displayed on each page")

def test\_TC\_093\_dynamic\_entry\_count\_filter(self):

driver = self.driver

driver.get("http://example.com")

initial\_entry\_count = len(WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector"))))

filter\_element = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "filter-selector"))) # Replace 'filter-selector' with the actual CSS selector for the filter element

filter\_element.send\_keys("valid filter")

filtered\_entry\_count = len(WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector"))))

self.assertNotEqual(initial\_entry\_count, filtered\_entry\_count, "The entry count should update dynamically to reflect the filtered entries")

def test\_TC\_094\_dynamic\_entry\_count\_remove\_filter(self):

driver = self.driver

driver.get("http://example.com")

filter\_element = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "filter-selector")))

filter\_element.send\_keys("valid filter")

filtered\_entry\_count = len(WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector"))))

filter\_element.clear()

filter\_element.send\_keys(Keys.RETURN) # Simulate removing the filter

final\_entry\_count = len(WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector"))))

self.assertNotEqual(filtered\_entry\_count, final\_entry\_count, "The entry count should update dynamically to reflect the original entries")

def test\_TC\_095\_invalid\_filter\_no\_update(self):

driver = self.driver

driver.get("http://example.com")

initial\_entry\_count = len(WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector"))))

filter\_element = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "filter-selector")))

filter\_element.send\_keys("invalid filter")

final\_entry\_count = len(WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector"))))

self.assertEqual(initial\_entry\_count, final\_entry\_count, "The entry count should not update with an invalid filter, and an appropriate message should be displayed")

def test\_TC\_096\_multiple\_filters\_update(self):

driver = self.driver

driver.get("http://example.com")

filter\_element1 = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "filter1-selector"))) # Replace 'filter1-selector' with the actual CSS selector for the first filter element

filter\_element1.send\_keys("valid filter 1")

filter\_element2 = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "filter2-selector"))) # Replace 'filter2-selector' with the actual CSS selector for the second filter element

filter\_element2.send\_keys("valid filter 2")

filtered\_entry\_count = len(WebDriverWait(driver, 10).until(EC.presence\_of\_all\_elements\_located((By.CSS\_SELECTOR, "entry-selector"))))

self.assertTrue(filtered\_entry\_count < initial\_entry\_count, "The entry count should update dynamically to reflect the entries matching the multiple filters")

def test\_TC\_097\_button\_style\_guide(self):

driver = self.driver

driver.get("http://example.com")

button = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "button-selector"))) # Replace 'button-selector' with the actual CSS selector for the button

self.assertEqual(button.value\_of\_css\_property('color'), 'expected-color') # Replace with the expected color from the style guide

self.assertEqual(button.value\_of\_css\_property('font-size'), 'expected-font-size') # Replace with the expected font size from the style guide

def test\_TC\_098\_icon\_style\_guide(self):

driver = self.driver

driver.get("http://example.com")

icon = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "icon-selector"))) # Replace 'icon-selector' with the actual CSS selector for the icon

self.assertEqual(icon.value\_of\_css\_property('color'), 'expected-color') # Replace with the expected color from the style guide

self.assertEqual(icon.value\_of\_css\_property('font-size'), 'expected-font-size') # Replace with the expected font size from the style guide

def test\_TC\_099\_text\_style\_guide(self):

driver = self.driver

driver.get("http://example.com")

text = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "text-selector"))) # Replace 'text-selector' with the actual CSS selector for the text

self.assertEqual(text.value\_of\_css\_property('font-size'), 'expected-font-size') # Replace with the expected font size from the style guide

self.assertEqual(text.value\_of\_css\_property('color'), 'expected-color') # Replace with the expected color from the style guide

def test\_TC\_100\_table\_layout\_desktop(self):

driver = self.driver

driver.get("http://example.com")

driver.set\_window\_size(1920, 1080) # Set window size to desktop resolution

table = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "table-selector"))) # Replace 'table-selector' with the actual CSS selector for the table

self.assertTrue(table.is\_displayed(), "The table layout should adapt appropriately to the desktop screen size")

def test\_TC\_101\_table\_layout\_tablet(self):

driver = self.driver

driver.get("http://example.com")

driver.set\_window\_size(768, 1024) # Set window size to tablet resolution

table = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "table-selector"))) # Replace 'table-selector' with the actual CSS selector for the table

self.assertTrue(table.is\_displayed(), "The table layout should adapt appropriately to the tablet screen size")

def test\_TC\_102\_table\_layout\_mobile(self):

driver = self.driver

driver.get("http://example.com")

driver.set\_window\_size(375, 667) # Set window size to mobile resolution

table = WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "table-selector"))) # Replace 'table-selector' with the actual CSS selector for the table

self.assertTrue(table.is\_displayed(), "The table layout should adapt appropriately to the mobile screen size")

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**NOTE:** Please ensure to replace the placeholders (e.g., http://example.com, CSS selectors, expected values) with actual values specific to your application.